Autonomous Virulence Adaptation Improves Coevolutionary Optimization

John Cartlidge and Djamel Ait-Boudaoud

***Abstract*—A novel approach for the autonomous virulence adaptation (AVA) of competing populations in a coevolutionary optimization framework is presented. Previous work has demonstrated that setting an appropriate virulence, v, of populations accelerates coevolutionary optimization by avoiding detrimental periods of disengagement. However, since the likelihood of disengagement varies both between systems and over time, choosing the ideal value of v is problematic. The AVA technique presented here uses a machine learning approach to continuously tune v as system engagement varies. In a simple, abstract domain, AVA is shown to successfully adapt to the most productive values of v. Further experiments, in more complex domains of sorting networks and maze navigation, demonstrate AVA's efficiency over reduced virulence and the layered Pareto coevolutionary archive.**

***Index Terms*—Autonomous virulence adaptation, coevolution, disengagement, genetic algorithms, machine learning, maze navigation, optimization methods, reduced virulence, sorting networks.**

I. Introduction

UNDER THE traditional genetic algorithm (GA) framework evolving solutions are evaluated using a static, predetermined, exogenous fitness function. By contrast, under the coevolutionary genetic algorithm (CGA) framework solutions are reciprocally evaluated against contemporaneous solutions that are themselves evolving; individuals receive an evaluation score based upon their relative performance and the genotype-fitness mapping varies over time. Measuring fitness in such a manner removes the necessity for operationally defining an objective function capable of transforming genotype space into fitness space in such a manner that evolution will progress. As a result, CGAs offer particular advantage in problem domains where an objective function is difficult to define with *a priori* knowledge. For example, consider chess: it is much easier to define the rules of interaction between strategies (the rules of chess) than it is to define a function capable of objectively evaluating each strategy (chess-playing ability).

CGAs come in three broad varieties: cooperative, competitive, and symbiotic. Cooperative CGAs are often utilized in domains that can be naturally decomposed into independent sub-components. Individuals are assessed through a series of collaborative groupings, forming complete solutions to a common goal [1], [2]. In contrast, competitive CGAs employ an antagonistic framework: evaluated through competition, individuals represent complete solutions that are gradually refined [3]–[5]. Finally, symbiotic (or symbiogenetic) CGAs enable increasingly complex individuals to evolve from simple sub-units via transmission of genetic material [6], [7]. Although species interactions are often mutualistic (benefiting all), other forms of symbiotic relationships may form; including parasitism (one benefits at the expense of the other), amensalism (one is harmed while the other is unaffected), and commensalism (one benefits without affecting the other) [8], [9].

Competitive CGAs can be decomposed into systems that contain one population engaged in self-play and systems with multiple, reproductively isolated, populations. Single-population systems are often used when the problem domain is symmetric. Examples include the game of tag, where contestants take turns to pursue and evade [4], backgammon [10], and Texas hold'em poker [11]. In each case, the challenge is identical whichever side a contestant plays. In contrast, 2-population systems are often favored when the problem domain is asymmetric; for example, competitions between list-sorting networks and unsorted lists [3], or pursuit and evasion contests with fixed roles [12]. Such domains naturally fit the 2-population framework; populations can take different genetic (and phenotypic) representations and evolve toward different objectives.

In asymmetric contests, there is often one side of particular interest. For example, in [3], evolving quality sorting networks is the primary objective; not the lists they sort. Common terminology describes the population we are interested in as *learners* (networks learn to sort) and the opposing population as *teachers* (the lists act as a teaching set for networks). Alternatively, using the biologically inspired terminology of parasitism, the primary objective population (networks) are *hosts*; the competing population (lists) are *parasites*.

This paper focuses *exclusively* on multiple population competitive CGAs, in particular 2-population systems.[1] One of the problems affecting these systems is *disengagement*: occurring when one population easily outperforms the other [13]. We present autonomous virulence adaptation (AVA), a novel technique designed to reduce the likelihood of disengagement, and demonstrate its efficacy in two complex domains.

Manuscript received February 9, 2009; revised June 27, 2009, November 25, 2009, February 23, 2010, and June 23, 2010; accepted July 29, 2010.

The authors are with the School of Computing, Engineering and Physical Sciences, University of Central Lancashire, Preston PR1-2HE, U.K. (e-mail: john@john-cartlidge.co.uk; djamel.ait-boudaoud@port.ac.uk).

Digital Object Identifier 10.1109/TEVC.2010.2073471

[1]Henceforth, the term CGA will be used to represent 2-population competitive coevolutionary GA systems.

This paper is organized as follows. Section II outlines common CGA pathologies and remedies, including disengagement and the reduced virulence (RV) technique. Harnessing the pertinent knowledge from this literature review, Section III introduces AVA, a technique capable of adapting RV values during run-time in response to variations in system asymmetry. In Section IV, the contrived greater than (GT) domain is used to determine robust parameter settings for AVA. Using these settings, experiments on sorting networks (Section V) and maze-navigating robots (Section VI) demonstrate the efficacy and versatility of AVA; requiring less domain knowledge than RV and lower computational cost than LAPCA, a standard Pareto archiving technique. Section VII concludes with an overall performance summary and comparison of AVA vs. alternative techniques, LAPCA and RV.

II. Competitive Coevolutionary Dynamics: A Review of the Major Pathologies and Remedies

CGAs suffer from several pathologies. To combat these, a suite of techniques have been developed. In this section, we give the common pathologies and remedies a cursory *aperçu*. Coevolving populations are denoted by $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ throughout.

A. Pathologies

1) *Overspecialization:* It occurs when coevolving populations become "too focused." By exploiting the idiosyncratic weaknesses of opponents, individuals may evolve in an unexpected (perhaps unwanted) direction; potentially leading to brittle solutions that are unable to generalize [14], [15].

Overspecialization, or *focusing*, is associated with the traditional machine learning problem of over-fitting; occurring when an algorithm adapts so well to a specific training set that the learning model treats noisy idiosyncrasies in the data as meaningful. In CGAs, overspecialization may be avoided by maintaining diverse training and rule sets ($\boldsymbol{P}_1$ and $\boldsymbol{P}_2$).

2) *Mediocre Stability:* CGAs may stabilize at a suboptimal equilibrium [10]; analogous to convergence at a local optimum in evolution. Interspecific collusion is often cited as the cause of mediocre stability [16]. For example, consider chess players taking alternate turns to play white. Players can improve reproductive success by striving to win; or they can throw every game they play black: if an opponent does the same, collusion guarantees both 50% of victories. While the first scenario leads to progress, the second results in mediocre stability that can be difficult to escape.

Intraspecific collusion is also possible [17]. By diversifying into separate niches, each focusing on a different subset of the full problem, a population may approach a mediocre stability of sub-standard, brittle individuals. However, while individuals are themselves brittle, the population as a whole *may* contain information necessary to produce generalist strategies. As [18] discusses, when using (co)evolution for learning, information from the whole population should be retained. Integrating these strategies, however, is non-trivial.

Arising as a result of collusion between over-specialized individuals, mediocre stability is related to the problem of overspecialization [17].

3) *Disengagement:* Consider coevolution as a coupled dynamical system with populations evolving over a dynamic fitness landscape fluctuating in response to perturbations from the other (see discussion of *NKC* landscapes in [19]). Then, it is this *interactive* dynamic that drives the selection pressure between populations, continually eroding any adaptive advantage (the Red Queen Effect [20]). If the system decouples, populations no longer perturb each other and any means of relative fitness assessment is eradicated. The result is evolutionary drift. In this context, disengagement is synonymous with system decoupling and occurs when one population easily outperforms the other.

Let x_i be the competitive score of individual i in population $\boldsymbol{P}$ and $\bar{x}$ be the mean population score, then populations disengage when

$$\sum_{i \in \mathbf{P}} (x_i - \bar{x})^2 = 0. \tag{1}$$

If variance in fitness scores is zero it becomes impossible to discriminate between individuals according to ability; the selection gradient disappears and the coevolving populations begin to stagnate. The result is a stymied system that is left to flounder aimlessly.

Disengagement has several idiosyncratic distinctions. When $\boldsymbol{P_1}$ reaches optimality before $\boldsymbol{P_2}$, *asymmetric* disengagement occurs: $\boldsymbol{P_1}$ mutants are selectively punished, while $\boldsymbol{P_2}$ has no gradient. In [8], symbiotic mutualism becomes commensalism once $\boldsymbol{P_1}$ finds a complete solution: $\boldsymbol{P_1}$ becomes a near-perfect host while $\boldsymbol{P_2}$ drifts toward smaller parasite-like individuals.

Effective disengagement occurs when noise (due to stochastic selection or evaluation) is greater than the true fitness signal. Although real valued competitions (e.g., [5]) enable a fitness gradient to remain when $\boldsymbol{P_1}$ outperforms $\boldsymbol{P_2}$, there is a threshold under which performance differences are overwhelmed by noise.

Finally, when populations disengage along a subset of (multiobjective) dimensions, *dimensionwise* disengagement occurs. This can produce focusing; hence, "disengagement will generally lead to overspecialization" [21].

Although the term "disengagement" has entered the literature relatively recently [22], [23], the phenomenon *has* previously been recognized. Coevolutionary engagement has been described as: "maintaining a gradient for selection" (or, conversely, disengagement as "loss of gradient") [14], "coevolving an ideal training set" [24], "maintaining learnability" [16], and "providing pedagogical stepping stones" [25].

4) *Cycling:* Several factors cause cycling. By assessing individuals against contemporaneous opponents, the selection pressure for retaining adaptations acquired for historical competitors may be removed, allowing previous adaptations to be lost. The resulting lack of momentum can lead to repeated cycles of discovery, loss, and rediscovery [12], [25]. Intransitive superiority relationships between competitors, where $A > B$, $B > C$, and $C > A$ can naturally lead to cycling [14], [22]. Random walks through strategy space may also lead to repetition, or cycling, though one that is aperiodic and unpredictable in nature [26]. Repeated arms-races and crashes,

such as those caused by disengagement, can trap a system in repetitive cycles [27]. Finally, populations may overspecialize to occupy a subspace which is optimal against contemporary competitors, but brittle to attack from other regions of space. Thus, the system may oscillate between low fitness solutions, resulting in a form of system level mediocre stability [16], [28].

In practice, it may be impossible to neatly disentangle CGA pathologies into discrete subsets. Nevertheless, categorization can aid in the development of a systematic methodology for designing effective remedies. The reader should note, however, that while effort has been made to make this category list exhaustive, it is entirely possible that other pathologies may exist.

B. Remedies

Techniques for combating CGA pathologies are summarized here. This is not intended as an exhaustive compilation, but rather an overview of the predominant methods and the challenges they are designed to overcome.

1) *Diversity Maintenance (DM):* DM is a technique primarily used to combat overspecialization. The literature is littered with DM (or niching) techniques (for a review, see [29]). Approaches include standard [30] and deterministic [31] crowding (restrict competition to genetically similar individuals), spatial embedding (restrict competition by spatial locality, first used in CGAs by [3]), and islands models (split populations into multiple demes that only interact through migration, originally [32]).

Fitness sharing (first proposed by [33], extended by [34]) is perhaps the most ubiquitous DM technique. In CGAs, fitness sharing comes in the forms of "competitive fitness sharing" [25], "implicit sharing" [35], and "resource sharing" [24]. The details of each are similar.

Resource sharing encourages niching; individuals are rewarded for beating opponents that few others can. Opponents are treated as a commodity or resource. Rather than gain a fitness point for each victory (simple fitness), one fitness point is shared among all competitors that beat a particular individual. Thus, competitors are rewarded less for how many opponents they beat and more for whom they beat.

2) *Archiving:* Used to encourage evolutionary momentum, archiving techniques trace a direct lineage back to elitism (originally [30]), where each generation's best competitor is preserved (first used in coevolution by [4] and [5]). Elitism was extended to multiple generations, [36], before [25] introduced the "Hall of Fame" (HoF), the first true coevolutionary archive. Each generation, population elites are added to the HoF. Individuals compete against a sample of competitors drawn from both the contemporary opponent population and HoF.

Preserving an evolutionary memory, the HoF is designed to stop cycling. Rapidly growing, however, the archive becomes computationally expensive. For efficiency, one extension adds only unbeatable individuals to the archive [37]. More recently, an alternative technique was developed to combat disengagement by storing individuals that outperform contemporaries in a "test bank" for later use [38]. In this way, the level of challenges is managed and genetic discoveries are efficiently reused.

3) *Pareto Coevolutionary Archive (PCA):* An alternative DM technique, PCAs preserve useful adaptations by explicitly treating the performance against each opponent as a dimension for optimization [6], [11]. Let $\boldsymbol{a} = (a_1, \ldots, a_n)$ denote the scores of individual A against each of $n \in \boldsymbol{n}$ opponents (objective dimensions) and $\boldsymbol{b} = (b_1, \ldots, b_n)$ denote B's scores against the same $\boldsymbol{n}$ opponents; then A Pareto dominates B on $\boldsymbol{n}$ *if and only if* A performs at least as well as B in all dimensions and better than B in one

$$A \overset{\boldsymbol{n}}{\succ} B \iff \forall n \in \boldsymbol{n} : a_n \geq b_n \quad \wedge \quad \exists n \in \boldsymbol{n} : a_n > b_n. \tag{2}$$

Individuals compete against all others in a set of pairwise competitions. Dominated individuals are destroyed, non-dominated preserved.

PCAs have matured from the early "Pareto optimality" algorithm [39] to the conservative DELPHI [21], the theoretically sound but impractical IPCA [40], and finally the usable LAPCA, capable of reliable performance without over sacrificing efficiency [41].

LAPCA uses a separate generator $G(\boldsymbol{P}_l, \boldsymbol{P}_t)$ and archive $A(\boldsymbol{A}_l, \boldsymbol{A}_t)$. G can take the form of any search process consisting of two populations: "learners," $\boldsymbol{P}_l$, and "tests," $\boldsymbol{P}_t$. Each generation, G, creates offspring which are submitted to A and stored if *useful*. Learner $l \in \boldsymbol{P}_l$ is useful when compared to the learner archive, $\boldsymbol{A}_l$, if non-dominated (2) and unique (does not have identical outcomes) on the test archive, $\boldsymbol{A}_t$. If there is a current test $t \in \boldsymbol{P}_t$ that makes l non-dominated and unique, then both l and t are added to $\boldsymbol{A}_l$ and $\boldsymbol{A}_t$, respectively. To avoid $\boldsymbol{A}_l$ and $\boldsymbol{A}_t$ bloating to an unmanageable size, once useful offspring have been added, archives are updated subject to the following process: 1) separate $\boldsymbol{A}_l$ into N Pareto layers such that all members of layer $\boldsymbol{n}_i$ dominate members of layer $\boldsymbol{n}_{i+1}$, then remove learners outside of the first N layers, and 2) preserve tests in $\boldsymbol{A}_t$ that distinguish learners (force a different outcome for any two learners) within each layer, and between contiguous layers, of $\boldsymbol{A}_l$. For details of this process, refer to [41].

HoF and LAPCA were compared through the evolution of neural network controllers of Pong players [42]. While there was no significant difference in performance, LAPCA maintained a smaller archive and so improved efficiency.

By maintaining an archive of non-dominated individuals, PCAs ensure that potentially useful adaptations are not lost, making it unlikely for the population to overspecialize along one dimension. Further, with dominance hierarchies potentially overcoming problems of instransitivity and the archive maintaining evolutionary memory, LAPCA encourages monotonic progress and reduces the possibility of cycling.

4) *Simple Fitness Transformations (SFT):* Several approaches using SFT have been proposed to alleviate disengagement. SFT is considered here as simple non-linear transformations of an individual's competitive fitness. Unlike PCAs or resource sharing, an SFT of individual i does not take account of competitive assessments in which i does not take part. Since SFT does not require additional assessments over

and above those necessitated by the standard coevolutionary framework, it is efficient.

Coevolutionary systems are often asymmetric, with populations differing either genetically (in terms of encoding) or behaviorally (in terms of goal strategy). Such asymmetry may result in an inherent advantage for one population [37]: "with many two population coevolutionary domains [there is] an intrinsic asymmetry in the difficulties faced by the two populations" [39]. It is easier to evade than pursue [43], to be an input list than a sorting network [13], to be a maze than a maze-navigator [44], and to be an initial condition than a classifier [45]. These biases encourage and exacerbate disengagement. However, they are not the exclusive cause. In the absence of bias, stochastic fluctuations can result in a (temporary) fitness advantage for one population [14].

To alleviate the problem of system bias leading to disengagement in the density classification task, the "entropy measure" [24], later adapted as the Φ function [45], was introduced. Having an inherent advantage, initial conditions (ICs) become increasingly difficult to classify as density, $\Delta(IC)$ approaches 0.5. To encourage challenging but not unclassifiable ICs, fitness is transformed such that

$$\Phi(IC) = \begin{cases} 0, & \text{if classified} \\ |\Delta(IC) - \frac{1}{2}|, & \text{otherwise} \end{cases} \quad (3)$$

the Φ function performs well, but is domain specific and not easily generalized.

The "phantom parasite" (PP) was the first domain general SFT designed to counteract disengagement [25]. PP reduces the fitness of unbeatable competitors. Fitness, $f(x)$, of individual, x, winning all, N, competitions is transformed such that $f(x) = \frac{N}{N-1}$. Individuals that lose at least one competition are unaffected.

More recently, the reduced virulence (RV) technique has been introduced by the authors [13], [23]. RV endeavors to rein in a population that inherits an advantageous bias: rather than reward individuals that thrash a competitor, RV favors individuals that are lenient. Perhaps counter-intuitively, RV encourages accelerated progress by handicapping the most successful solutions. Fitness, $f(x_i, v)$, of solution, i, in population, $\mathbf{P}$, is calculated using i's competitive score, x_i, and population virulence, v, according to the RV equation

$$\forall i \in \mathbf{P}: \qquad f(x_i, v) = \frac{2x_i}{v} - \frac{x_i^2}{v^2} \quad (4)$$

where $0.5 \leq v \leq 1.0$ and $0 \leq x_i \leq 1$. Notice that when $x_i = v$, $f(x_i, v)$ is maximized. Also, when $v = 1$, $f(x_i, v)$ is monotonically increasing and (4) reduces to the canonical evaluation: *reward all wins*. Previous work has demonstrated that RV can accelerate progress significantly when evolving sorting networks [13].

Although inspired by virulence in biological host-parasite systems, RV is a loose analogy of its natural counterpart. Here, virulence is defined as the SFT formalized in (4). The biological analogy holds such that when virulence is reduced, individuals are rewarded for beating competitors *less severely*; roughly approximating a biological reduction in "parasite-mediated morbidity and mortality in infected hosts" [46].

Following the rationale of RV, the "soft parasites" (SP) approach was proposed, where SP fitness is "the standard deviation of the fitness foregone by the hosts encountering it" [47]. In the game of Tartarus, host agents and parasite board configurations were coevolved. "Young" agents—those that had survived less than one evaluation—were often poor quality and thus of little worth in assessing the ability of boards to generate a fitness gradient for agents. This produced a mediocre stable state with persistent low quality boards and agents. Performance improved by computing board fitness from encounters with "older" agents only. Thus, "age" was introduced as a tunable parameter to overcome the vulnerability of SP to low quality mutants [47].

In summary, RV is a domain independent technique that can be tuned to combat specific biases. As such, RV has greater versatility than PP, SP, and Φ. Additionally, by setting a high value of $v < 1.0$, the RV transform approximates PP and so is a generalization of this technique. In direct comparison, RV was shown to significantly outperform PP in the domain of sorting networks [13]. Further, by setting $v = 0.5$, the RV transform approximates the Φ function, (3). Thus, RV covers the previous SFTs designed to combat disengagement.

However, RV raises the question of what value of v to choose. If challenges are too hard then nothing is learned, too easy and there is no challenge [16]. Selecting an inappropriate v will hinder performance [48]. To address this problem, Section III develops a new technique capable of adapting virulence autonomously.

III. Autonomous Virulence Adaptation

Section II discussed how intrinsic asymmetry within coevolutionary systems can lead to disengagement. To counter this, the RV technique was introduced; utilizing a virulence parameter, v, that should be set to a value that ideally counterbalances asymmetric bias. However, since in real systems it is difficult to know *a priori* what this bias will be, or whether it will remain fixed, selecting an appropriate value of v is largely a matter of guesswork. As such, it would be advantageous to be able to automatically update v intelligently during runtime.

Previously, a method for human controlled steering of virulence has been introduced [49], allowing users to manually update virulence during runtime via a graphical interface. However, such steering can be problematic, causing human fatigue during long or repeated simulations. Here, we introduce a novel technique for AVA that is domain general, simple to implement, and versatile.

First presented in [50], AVA makes use of the Widrow-Hoff "delta" rule [51] to update virulence. This general learning procedure minimizes the error between an actual system output and a target output. By defining a proxy for target virulence, AVA updates v each generation by minimizing the error between the current virulence and the target proxy.

A. Delta Rule

Popularized by [52], one of the simplest update rules in machine learning is the delta rule. Originally developed as an

TABLE I
TABLE OF SYMBOLS

Symbol	Description
β_i	Mutation bias for population i: probability mutated bit = 1
RV(v_i, v_j)	Reduced virulence: v_i, v_j fixed
v_i	Virulence value of population i
MAX	Maximum virulence, RV(1, 1)
AVA(α, τ, μ, v_0)	Autonomous virulence adaptation: v updated each generation
α	Rate of change of virulence
τ	Target mean relative score
μ	Momentum coefficient
v_0	Initial virulence value

error minimization rule for updating the weights of neurons in simple feed forward networks with *no* hidden layers, the delta rule was shown to guarantee learning [51]. This mechanism was later extended to facilitate back-propagation learning in feed forward networks *with* hidden layers: the "generalized delta rule" [52].

The delta rule compares a network's output with the desired output for a given input. If there is no difference, then no learning takes place. Otherwise, the weights of the network are changed to reduce this difference. For linear units, the square of the differences between the actual and desired output values is minimized. To prevent oscillations, a momentum term is used: a constant value that determines the effect of past weight changes on the current direction of movement in weight space.

Following [53], let A_t be the actual output at time t and A_{t+1} be the actual output on the following time step, then

$$A_{t+1} = A_t + \Delta_t \tag{5}$$

where Δ_t is determined by the product of a learning rate, α, and the difference between A_t and the target output, T_t

$$\Delta_t = \alpha(T_t - A_t). \tag{6}$$

If the target remains constant, (5) and (6) give asymptotic convergence of A_t to T_t, at a speed determined by α. However, if the target is not fixed, A_t may oscillate around T_t. To dampen these oscillations, a momentum term μ can be added, such that (6) becomes

$$\Delta_t = \mu\Delta_{t-1} + \alpha(1 - \mu)(T_t - A_t). \tag{7}$$

Notice that when $\mu = 0$, (7) reduces to (6); when $\mu = 1$, then $\forall t : \Delta_t = \Delta_0$.

B. Deriving the AVA Equations

To adapt virulence during run-time, an update rule based on the delta rule is used. However, since the target output (virulence) at each time step is unknown, we use an associated variable: the target mean relative score of the population, τ. Using the difference between τ and $\overline{x}_t$, the actual mean relative score of the population at time t, substituting into (6), we get

$$\Delta_t = \alpha(\tau - \overline{x}_t) \tag{8}$$

where α is the rate of change of virulence, v. Adding a momentum coefficient, μ and letting v_t and v_{t+1} be the virulence at time t and time $t + 1$ respectively, substituting into (5) and (7) we get the AVA equations

$$v_{t+1} = v_t + \Delta_t \tag{9}$$

where

$$\Delta_t = \mu\Delta_{t-1} + \alpha(1 - \mu)(\tau - \overline{x}_t) \tag{10}$$

and $\Delta_0 = 0$. Notice that if $\mu = 0$, then (10) reduces to (8), i.e., no momentum; if $\mu = 1$, then $\forall t : \Delta_t = \Delta_0$, i.e., fixed $v = v_0$. AVA's three parameter settings must fall in the range, $0 \leq \alpha, \mu, \tau \leq 1$. For a summary of symbols and their meanings, refer to Table I.

In Section IV, the greater than (GT) game is used as an experimental platform to test the performance of AVA in a controlled environment. Robust parameter values are suggested that enable AVA to be used off-the-shelf as a domain-general technique. In Sections V and VI, these values are used to test AVA in two complex domains.

IV. GREATER THAN GAME

The AVA equations introduce three new parameters: rate of change, α, momentum, μ, and target, τ (see Table I). Here, we use the GT game to identify robust values for these parameters. Using mutation bias, β, as a controllable proxy for asymmetry, the GT game can simulate a broad spectrum of real world coevolutionary domains with large or small, fixed or varying, asymmetry. Evidence for a robust parameter set is gathered by evolving AVA parameters across a variety of GT simulations.

The reader should be aware that GT has no natural asymmetry since populations have identical genetic representations and performance evaluation. To introduce asymmetry, mutation is varied between populations. This mutational bias is used as a surrogate for distinct representations that occur in more realistic systems and offers the advantage of being easily tuned. Mutation bias has been used as a proxy for genetic representations elsewhere (e.g., [21], [41]).

A. GT Problem Definition

GT was initially developed to demonstrate how CGAs can fail even in trivial domains [14]. The game is very simple: bit-strings of length L compete in pairwise contests, comparing which has the greater number of bits set to 1. Let $|i|$ denote the sum of bits in the bit-string of individual i. Then, i receives a normalized score, $x(i, \mathbf{S})$, against opponent sample, $\mathbf{S}$, as follows:

$$x(i, \mathbf{S}) = \frac{1}{|\mathbf{S}|} \sum_{j \in \mathbf{S}} g(i, j) \tag{11}$$

where

$$g(i,\, j) = \begin{cases} 1, & \text{if } |i| > |j| \\ 0.5, & \text{if } |i| = |j| \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

the goal of GT is to evolve a string with all bits set to 1; i.e., when $\exists i \in \mathbf{P} : |i| = L$. Disengagement occurs when all individuals within a population either score 1 or 0, that is

$$\overline{x} = \frac{1}{|\mathbf{P}|} \sum_{i \in \mathbf{P}} x(i, \mathbf{S}) = 1 \text{ or } 0. \tag{13}$$

B. GT Coevolution

Two reproductively isolated populations (of size 25) coevolve for N_{gen} generations. Populations are labeled "hosts" and "parasites." Bit-strings ($L = 100$) are initialized to 0. Each generation, individuals are assessed by playing a sample of five opponents from the opposing population. Tournament selection is used (size 2); winner reproduces asexually. Mutation occurs at each bit with probability $m = 0.005$. For each mutation event, the current bit is replaced by a 1 or 0 with probability β and $1 - \beta$, respectively. Each run, $\beta_{host} = 0.5$. Bias is controlled by varying β_{par}. Thus, when $\beta_{par} > 0.5$, there exists an asymmetry in favor of parasites. All coevolutionary runs are repeated N_{run} times.

C. Evolving AVA Parameters

1) *Experimental Setup:* A population of ten individuals (each an AVA parameter set) evolve over 30 generations. Each generation the fittest individual is copied unchanged into the following generation (elitism). The remaining individuals are sexually selected using single-point crossover; tournament size 2. The probability of parameter mutation decreases linearly over time from 0.5 to 0. When it occurs, mutation increments the chosen parameter by a uniformly distributed value in the range $\pm\varepsilon$, where ε decreases linearly over time from 0.25 to 0. Parameter values are truncated at 0 and 1.

Individual parameter sets are evaluated using performance over $N_{gen} = 500$ generations of GT coevolution (Section IV-B). To encourage robustness, assessment consists of a set of GT runs using a combination of bias and mutation settings. During fixed bias assessment β_{par} remains constant. During variable bias assessment β_{par} varies over time. The fitness, f_p, of each parameter set, p, over N_{run} assessments is calculated as follows:

$$f_p = \sum_{i=1}^{N_{run}} h_i + g_i - 100 d_i \tag{14}$$

where h_i is the highest objective host score, g_i is the number of generations containing a host with maximum objective score of $L = 100$, and d_i is the total number of disengaged generations during each run, i. Equation (14) rewards parameters that quickly find and maintain a host solution with an objective score of 100, while disproportionately punishing parameters that allow disengagement to occur.

Fixed bias assessment is used for two sets of experiments, A ($N_{run} = 6$) and B ($N_{run} = 3$). GT settings

$$\text{A: } \beta_{par} = \{0.5, 0.75, 0.95\}\,,\, m = \{0.01, 0.015\}$$
$$\text{B: } \beta_{par} = \{0.05, 0.5, 0.95\}\,,\, m = \{0.01\}\,.$$

Variable bias assessment is used for three sets of experiments C–E. Under each condition, $N_{run} = 4$ and $m = 0.01$. Let β_g^x denote a change of bias to value x at generation g and β_{g-h}^{x-y} denote a linear change in β between generations g and h from value x to y. Then, bias profiles are

$$\text{C: } \left\{\beta_0^{0.05}, \beta_{50-450}^{0.05-0.95}, \beta_{450-500}^{0.95-0.5}\right\}$$
$$\text{D: } \left\{\beta_0^{0.01}, \beta_{20}^{0.99}, \beta_{450}^{0.01}\right\}$$
$$\text{E: } \left\{\beta_0^{0.05}, \beta_{25}^{0.95}, \beta_{120}^{0.05}, \beta_{140}^{0.95}, \beta_{310}^{0.05}, \beta_{400}^{0.95}\right\}.$$

2) *Results:* Conditions A–B were repeated 30 times and C–E 15 times. Table II displays the mean value (and 95% confidence interval) of each parameter contained within the fittest individual of the final generation of each run.

Across all conditions, target, τ, tends to a value in the region of 0.6. When bias is fixed (A–B) τ tends to a significantly higher value than when bias is varied through large discontinuous jumps (D–E). When bias changes gradually over time (C) τ tends to an intermediate value. This result is intuitive. Since an increase in τ induces more aggressive increases in population virulence,[2] higher values of τ succeed when the coevolutionary system is unlikely to be perturbed by a sudden shift in advantage. Conversely, adopting lower values of τ results in lower population virulence. This reduces the speed of evolution and maintains greater engagement, enabling the system to better endure sudden perturbation.

Reaction rate, α, follows a more pronounced trend. When bias is fixed (A–B) α tends to significantly lower values than when bias alters suddenly (D–E). Again, a gradual change in bias (C) results in an intermediate value. This is to be expected. Since α controls the speed at which virulence varies, greater values of α allow the system to react more quickly to sudden changes in bias. However, high values of α can also induce rapid virulence fluctuations, potentially leading to system instability. For this reason, when bias remains constant, lower values of α prevail.

Finally, momentum, μ, tends to converge to a value in the region of 0.3 under all conditions. There is no significant difference between fixed and variable bias runs. This suggests that μ is robust to changes in bias.

D. Selecting Robust AVA Parameters

Upon first consideration, it may appear that replacing two RV parameters with three AVA parameters introduces some extra work unnecessarily. However, this is not the case. First, by adapting v, AVA is able to perform well in systems where asymmetrical bias changes over time. This ability makes AVA a more powerful technique than RV. Second, it will be shown

[2] As τ is increased, the virulence of both populations will tend to increase also.

TABLE II
EVOLVED AVA PARAMETERS IN THE GT GAME USING FIXED AND VARIABLE BIAS (MEAN ± 95% CI)

Experiment	α	μ	τ
A: β_{par} fix	0.10 ± 0.07	0.25 ± 0.08	0.63 ± 0.02
B: β_{par} fix	0.11 ± 0.06	0.34 ± 0.10	0.66 ± 0.04
C: β_{par} var	0.29 ± 0.15	0.35 ± 0.13	0.62 ± 0.03
D: β_{par} var	0.37 ± 0.18	0.20 ± 0.09	0.59 ± 0.02
E: β_{par} var	0.43 ± 0.19	0.26 ± 0.13	0.58 ± 0.02

that there are standard AVA values that perform well in a variety of domains. Unlike RV, AVA settings can be used off-the-shelf, assuming no prior domain knowledge.

Here, we select AVA settings suggested by the evidence gathered in Section IV-C, where a variety of GT bias profiles were used to simulate asymmetry across a broad spectrum of real world coevolutionary problems. To demonstrate AVA's generality and robustness, these settings will be used throughout this paper and tested in a variety of domains.

To encourage progress it is necessary to set $\tau > 0.5$. When $\tau = 0.5$, an increase in the virulence of one population causes an identical decrease in the virulence of the other, thus removing the capacity to have two high-virulence populations. When $\tau < 0.5$, the virulence of both populations will tend toward 0.5 and retard progress. Conversely, if τ is set too high disengagement becomes more likely. From Table II, results suggest that $\tau \approx 0.6$ is robust, performing well under many bias conditions. Selecting a value slightly lower than this exercises caution. Set $\tau = 0.56$.

Reaction rate, α, controls the speed at which AVA is able to update virulence. When system bias changes slowly α should be set at a lower value than when bias changes rapidly. Table II confirms this. Intuitively, it is likely that many problem domains have a slowly changing bias, thus making a low value of α preferable. Further, since erratic virulence fluctuations can lead to system instability, a very low value of α should be selected. Set $\alpha = 0.0125$.

Finally, guided by Table II, set $\mu = 0.3$.

E. Mapping the Virulence Sweet Spot

Here, we identify the virulence "sweet spot" for GT: values of v that are the most productive given a particular system bias. While this information is not useful in its own right—given the contrived nature of the problem we cannot draw general conclusions from this data—it does allow us to identify a "target" value of v for a given system bias in this one domain. This target is used in Section IV-F to evaluate AVA's ability to adapt virulence appropriately.

AVA offers a solution to the problem of setting virulence. However, without understanding the virulence required for a specific task, it is impossible to know whether or not a self-adapting technique is following a desirable trajectory. The target sweet spot identified in this section offers the possibility of objective evaluation: can adaptation hit the target across a range of biases? Armed with this information, we can be confident whether or not virulence is being adapted in a suitable manner before testing the system in more practical domains (Sections V and VI).

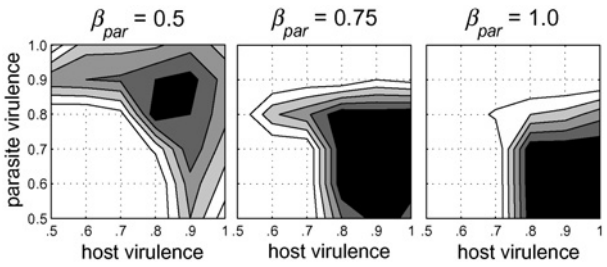

Fig. 1. Contour plots showing the RV sweet spot (successful runs with no disengagement, identified by the black regions) in the GT game under varying levels of bias. With no bias ($\beta_{par} = 0.5$, left) we see a symmetric plot, with v values between 0.8 and 0.9 giving best performance. As β_{par} is increased to 0.75 (center) and then 1.0 (right), we see that v_{host} must be increased and v_{par} decreased.

1) *Experimental Setup:* We follow the GT coevolutionary setup described in Section IV-B, with $N_{run} = 30$ and $N_{gen} = 750$. Coevolution is run using a range of fixed virulence settings. Let RV(v_{host}, v_{par}) denote host virulence v_{host} and parasite virulence v_{par}. Also, let MAX denote the canonical setup of RV(1, 1).

2) *Results:* The contour plots of Fig. 1 show the number of "successful" GT runs (0–5, . . . , 26–30, from light to dark). Success implies that the GT goal has been reached: $\exists i \in \mathbf{P}_{host} : |i| = 100$. However, since we are interested here in avoiding disengagement, runs that reach the GT goal temporarily, but then disengage and fall away from this goal, are not counted as successful. Thus, in this section we consider a successful run as one in which the GT goal is reached *and* there are no periods of disengagement (13). Settings with greater than 25/30 successful runs are shown in black.

As bias, β_{par}, increases in favor of parasites (left to right), the ideal values for v_{host} and v_{par} increase and decrease, respectively. This is an intuitive result: the greater the system bias, the more the advantaged population should be reined in. Interestingly, with no bias (left), the canonical setup, MAX, is not optimal; inducing disengagement in more than half of all runs. In comparison, RV(0.8, 0.8) produced no disengagement. However, there is a tradeoff here: low v reduces disengagement; high v accelerates evolution. Balancing these antagonistic constraints is the key to success. To obtain swift progress with minimum disengagement, ideal virulence pairs are located at the top-right hand corner of each of the sweet spots (black regions) shown in Fig. 1.

F. Autonomous Virulence Adaptation

Here, using AVA rather than RV, we repeat the experiment of Section IV-E to see how AVA adapts virulence in the GT game under different bias levels. To avoid immediate disengagement when bias is high, for the initial $I = 4$ generations, virulence is free to rapidly change by replacing (10) with $\Delta_t = (0.5 - \overline{x}_t/t)$. AVA settings for each run are those selected in Section IV-D $\alpha = 0.0125$, $\tau = 0.56$, $\mu = 0.3$, $v_0 = 0.75$.

Fig. 2 shows the adaptation of v_{host} and v_{par} over time under differing bias conditions, β_{par}. Error bars show the 95%

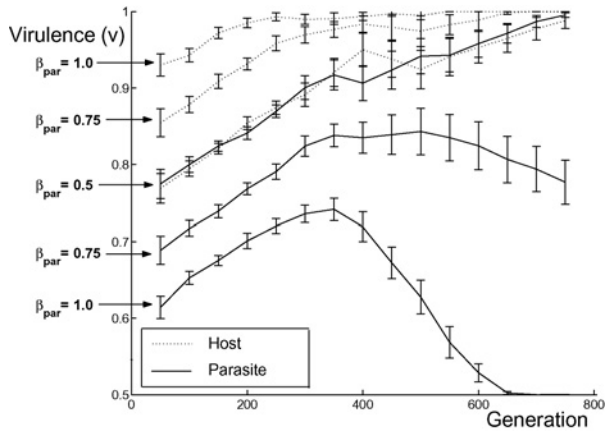

Fig. 2. Graph showing the 95% confidence of mean value of v_{host} and v_{par} over 30 runs of coevolution in the GT game using AVA. With no bias (β_{par} = 0.5) the virulence of both populations steadily increases to near maximum. As β_{par} is increased to 0.75 and then 1.0, v_{host} quickly approaches 1 while v_{par} approaches much lower values, near 0.75 and 0.5.

confidence interval of the mean over 30 runs. We see that for each bias condition, AVA initially adapts v_{host}, v_{par} toward the ideal virulence pairing at the top right of the sweet-spots shown in Fig. 1.

On each run, both populations discover successful individuals around generations 350–400, where success is determined as follows:

$$\exists i \in \mathbf{P}_{host},\, j \in \mathbf{P}_{par} : |i| = |j| = 100. \tag{15}$$

At this point, the GT game is solved and there can be no more progress. As a result, when $\beta_{par} > 0.5$, AVA decreases the value of v_{par}, ensuring system engagement whilst maintaining successful individuals.

These results demonstrate that AVA is capable of autonomously adapting toward GT's target virulence sweet spot.

G. AVA vs. RV

To evaluate the utility of AVA, a strict comparison is performed against fixed RV values across bias levels, $\beta = \{0.5, 0.6, \ldots, 1.0\}$. Table III shows the mean number of successful runs, (15), for each fixed value-pair of v. Each cell, $c(i, j)$, is calculated as follows:

$$c(i, j) = \frac{s(i, j) + s(j, i)}{2} \tag{16}$$

where $s(i, j)$ is the number of runs (of 180) that RV(v_i, v_j) succeeds. Since bias may favor either population, we are able to utilize the symmetry of results. The most successful combination is RV(0.8, 0.8) with 166 successful runs and 6,269 disengaged generations (DGs). The canonical setup, MAX, produced 79 successful runs and 68 900 DGs; a degree of magnitude worse. Thus, when system bias is unknown, it is unwise to assume the canonical setup is best.

In contrast, AVA significantly outperforms all fixed RV value pairs. Under the same conditions, AVA is successful in *every* run (180) and induces only 7 DGs. Using χ^2 to compare the number of successes and failures of AVA and RV(0.8, 0.8),

TABLE III
MEAN NUMBER OF SUCCESSFUL GT RUNS (MAX. 180) ACROSS ALL BIAS LEVELS USING RV VALUE, v, AND AVA

v	0.5	0.6	0.7	0.8	0.9	1.0	AVA
0.5	0	0	0	59	148	124.5	
0.6		0	0	101.5	150	128.5	
0.7			0	140	151	128.5	
0.8				166	149	126.5	
0.9					129	101	
1.0						79	
AVA							180

the null hypothesis, H_0 : {*proportion of successes of both binomial distributions is equal*}, is rejected with 99.9% confidence ($\chi^2_{(1)} = 14.56$, giving a one-tailed p value of $p < 0.0001$). The difference is primarily due to RV(0.8, 0.8) under-performing when both bias levels are low (β_{par}, $\beta_{host} = 0.5$). Under these conditions, v is not high enough to counter-act the rapid insertion of deleterious mutations each generation. Although the system remains engaged, it falls into a mediocre-stability such that

$$\nexists i \in \mathbf{P}_{host},\, j \in \mathbf{P}_{par} : |i| = |j| = 100. \tag{17}$$

Under these conditions, AVA is able to increase virulence toward $v = 1$ in both populations, effectively overcoming the low mutation rate.

In summary, AVA is capable of automatically updating virulence toward values known to be productive in the GT game. As a result, AVA outperforms all fixed virulence pairings when tested across all bias conditions. However, GT is a contrived domain and therefore uninteresting in isolation. To address this, in the following section we test AVA in a more practical domain of sorting networks.

V. SORTING NETWORKS

We test AVA in a complex domain of fixed length sorting networks. The challenge is to discover a network of comparisons that is able to sort input lists of a fixed length. In [3], coevolution was shown to outperform a standard evolutionary approach, discovering a network of 61 comparisons for inputs of length 16; only one comparison longer than the shortest network is known. In [54], coevolution set a new record, discovering a network of 45 comparisons to sort lists of length 13; one comparison less than the previous best. RV has been shown to accelerate coevolution in this domain [13].

A. Network-List Coevolution

We use the following genetic representation.

1) *Input List Sets:* Input sets contain 40 lists of length 13. Input lists consist of permutations of the natural numbers 1–13 and are encoded as an array of 13 integers. Each generation integer is mutated with probability 0.01. When it occurs, mutation swaps the integer with another integer in the list, randomly selected from a uniform distribution. Thus, mutation preserves lists as permutations of the first 13 integers. Each generation input list is macro mutated with probability 0.02:

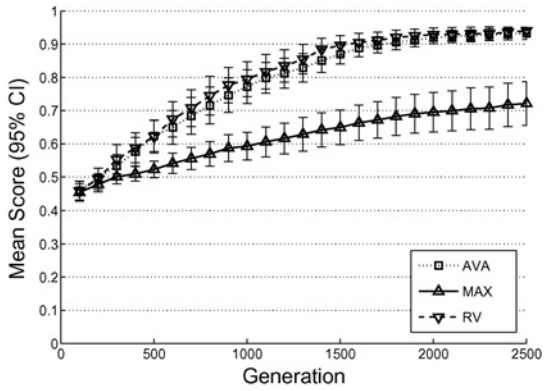

Fig. 3. Mean performance (±95% C.I.) of best coevolved sorting networks over 30 runs. Both AVA and RV significantly outperform MAX (the control).

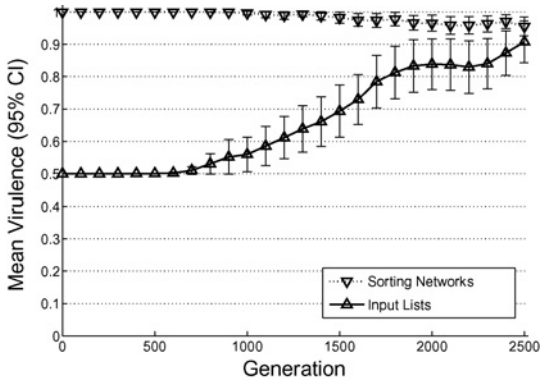

Fig. 4. Mean virulence (±95% C.I.) of coevolved sorting networks and input lists over 30 runs. Sorting networks initially have a high virulence of 1.0, with input lists having low virulence of 0.5. Over time the virulence of input lists gradually increases as the bias between populations falls.

lists are copied onto another list randomly selected (uniform distribution) from the full list set. Input lists are initially randomized.

2) *Sorting Networks:* Sorting networks contain 45 comparators. Comparators of the form (i_a,i_b) compare inputs i_a and i_b at indexes a and b. If $i_a > i_b$ and $a < b$ then the inputs are swapped. In this way networks aim to fully sort an input list into ascending order. Networks of 45 comparators are encoded as an array of 90 integers in the range $[1, 13]$, each representing the input list index to compare. Each generation comparison integer is mutated with probability 0.02. A value is selected at random from a uniform distribution in the range $[1, 13]$. Sorting networks are initially randomized.

3) *Evaluation:* Networks score in proportion to the number of lists that are fully sorted: score 1 if all 40 lists are sorted, 0 if no lists are sorted. List sets score 1 minus network score.

4) *Coevolution:* 25 networks and 25 list sets coevolve for N_{gen} generations (asexual reproduction, tournament size 5). Each generation network is evaluated against one list set (and *vice versa*). All runs are repeated N_{run} times. Let RV(v_n,v_i) denote fixed network virulence, v_n, and fixed input lists virulence, v_i. Also, set MAX = RV(1, 1) and AVA ($\alpha = 0.0125$, $\tau = 0.56$, $\mu = 0.3$, $v_0 = 0.75$). Notice that these are the same AVA parameter values established in Section IV-D for the GT game. While it is unlikely that these values are optimal for list sorting, the aim is to test the generality of AVA by using the same settings across various domains.

B. AVA vs. RV

Compare the performance of AVA and RV.

1) *Experimental Setup:* We follow the coevolutionary setup described in Section V-A, with $N_{run} = 30$ and $N_{gen} = 2500$. Coevolution is run under three conditions: MAX, RV(1, 0.75), and AVA.

2) *Results:* Fig. 3 displays the mean score (±95% C.I.) of the best network so far over 30 runs. Score is calculated by exhaustively evaluating the current fittest network against the set of all possible input lists. It should be noted that networks do not have exposure to this full set of input lists during their evolutionary learning.

RV(1, 0.75), henceforth denoted RV, maintains engagement and accelerates coevolution (outperforming MAX). In contrast, MAX induces prolonged periods of disengagement, effectively wasting evaluation cycles.

AVA performs similarly to RV and significantly outperforms MAX. Fig. 4 shows mean population virulence across all runs. Initially, AVA sets $v_n = 1.0$ and $v_i = 0.5$; correctly adapting to the fact that it is initially much easier to be a difficult list to sort than a successful sorting network. By generation 750, networks can sort nearly 70% of all input lists (Fig. 3). Reflecting this, AVA begins to increase v_i (Fig. 4). By generation 2000, $v_i \approx 0.85$ and $v_n \approx 0.95$. At this stage networks can sort $> 90\%$ of lists. With bias shifting in favor of sorters it is no longer necessary for lists to give networks an "easy ride."

C. AVA vs. LAPCA

We have demonstrated AVA outperforming MAX in the sorting networks domain. However, this does not tell us how AVA performs in relation to other techniques designed to improve coevolutionary optimization. We test this here by comparing AVA with LAPCA (Section II-B3), a PCA designed to provide monotonic progress by overcoming focusing (Section II-A1) and intransitivity (Section II-A4); and the mediocre stability (Section II-A2) or cycling (Section II-A4) that may result [41].

We test LAPCA using an archive size of $N = 5$ layers. Consider networks to be "learners" and list sets to be "tests." Preliminary runs showed the learner archive grows to an unmanageable size, $|\boldsymbol{A}_l| > 600$. To prevent this, $\boldsymbol{A}_l$ was constrained to a maximum size, $|\boldsymbol{A}_l|^{max} = 250$. Learners were rejected from $\boldsymbol{A}_l$ using a "pruning" process, similar to that developed by [11]. $\boldsymbol{A}_l$ is initially separated into N dominance layers. Starting with the highest dominance layer, $\boldsymbol{n}_1$, the number of learners, $|\boldsymbol{n}_i|$, in each layer, i, is summed. If the sum of learners in the current layer and all higher layers is less than the maximum, then preserve the current layer. More formally, preserve $\boldsymbol{n}_i$ if

$$|\boldsymbol{A}_l|^{max} < \sum_{j=1}^{i} |\boldsymbol{n}_j|. \tag{18}$$

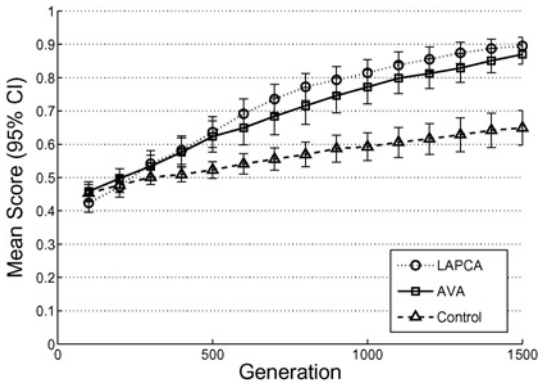

Fig. 5. Mean performance (±95% C.I., 30 runs) against generational time of coevolved sorting networks. Both LAPCA and AVA outperform the control on a generational basis. The difference between LAPCA and AVA is not significant at the 0.05 level.

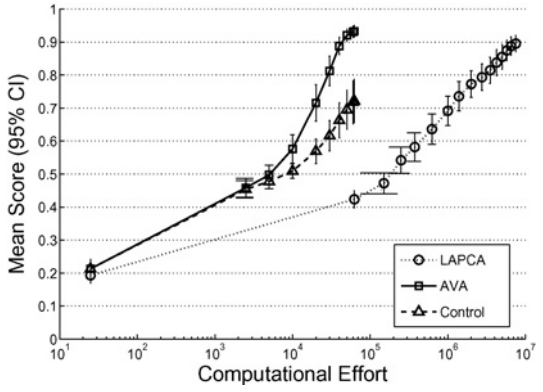

Fig. 6. Mean performance (±95% C.I., 30 runs) against computational effort (number of network evaluations, presented on a log-scale) of coevolved sorting networks. AVA significantly outperforms the control. In contrast, LAPCA requires an order of magnitude more effort to reach equivalent performance.

Following [11], for non-preserved layers find the closest rival for all learners—the rival on which l dominates least, or has the fewest winning dimensions—and store this number. Starting with $\boldsymbol{n}_N$, while $|\boldsymbol{A}_l| \geq |\boldsymbol{A}_l|^{max}$: sequentially remove (in random order) individuals that dominate their closest rival on only one dimension, then two dimensions, and so on.

This process constrains archive size, while preserving the most dominant learners. We use LAPCA ($N = 5$) pruned such that $|\boldsymbol{A}_l|^{max} = 250$. For direct comparison, the generator uses MAX with the same setup as Section V-A. Parents are selected from the archives with probability 0.05.

1) *Experimental Setup:* We follow the setup described in Section V-A, with $N_{run} = 30$ and $N_{gen} = 1500$. Coevolution is run under three conditions: MAX, AVA, and LAPCA.

2) *Results:* Fig. 5 shows the mean (±95% C.I.) performance of 30 runs of sorting network coevolution. For comparison, the previous results of AVA and MAX are reproduced from Fig. 3. It is clear that LAPCA outperforms the control (MAX) when compared using generational time: where confidence intervals *do not* overlap, the difference is statistically significant (t-test) with $p < 0.05$. While Fig. 5 suggests LAPCA has a slightly higher performance than AVA, results are not significant at the 0.95 level (t-test) at any generation.

LAPCA improves performance over generational time by maintaining an archive of useful learners and tests that prevent prolonged disengagement in the generator. In general, by generation 1000, archive sizes stabilize with $|\boldsymbol{A}_l| = |\boldsymbol{A}_l|^{max} = 250$ and $|\boldsymbol{A}_t| \approx 35$. Further runs were performed using AVA as a generator in combination with LAPCA (results not shown).

Over generational time, LAPCA outperforms the control (Fig. 5). However, this does not consider the computational cost of each mechanism. While AVA requires no additional assessments (over the control), the same is *not* true for LAPCA. In order to calculate dominance relationships for the archive, many additional assessments are required. If archive results are cached, it is necessary (at minimum) to assess each new learner against all contemporary tests and all archived tests; and each new test against all archived learners. Therefore, the number of additional LAPCA assessments, C, required each generation is

$$C = |\boldsymbol{P}_l|.|\boldsymbol{P}_t| + |\boldsymbol{P}_l|.|\boldsymbol{A}_t| + |\boldsymbol{A}_l|.|\boldsymbol{P}_t|. \tag{19}$$

For the current experiments, $|\boldsymbol{P}_l| = |\boldsymbol{P}_t| = 25$ for all runs. Under the control condition (and AVA), each generation there are 25 assessments. For LAPCA $C^{min} = 625$ and $C^{max} = 625 + 6250 + 1250 = 8125$ (with $|\boldsymbol{A}_l|^{max} = 250$, and assuming $|\boldsymbol{A}_t| \leq 50$). Thus, each generation, the computational cost of LAPCA is 1 to 2 orders of magnitude greater than AVA. This cost grows with $|\boldsymbol{P}_{l,t}|$ and $|\boldsymbol{A}_l|^{max}$ and becomes increasingly problematic in domains where the assessment process is complex.

Reconsidering results from the perspective of computational efficiency, Fig. 6 presents mean performance against effort (number of assessments; see [55] for "equal effort comparisons"). Using this metric, it is clear that LAPCA requires significantly more effort. This is compelling evidence to suggest that, in the sorting networks domain, at least, AVA is a practical alternative to LAPCA with much lower computational cost (given the configuration settings used here).

VI. Maze Navigation

To test whether the results from the previous section are representative, we assess AVA on another challenging problem: maze navigation. This domain offers more complexity than sorting and includes stochastic assessment. In addition, there is an asymmetric bias between populations (in both representation and goal difficulty), making disengagement likely to occur. As such, maze navigation is a problem domain suitable to test the efficacy of AVA.

A. Background

Maze navigation is a simplification of the more general problem of real-world navigation by autonomous robots. Simulated robots are placed in simple 2-D grid worlds (mazes) containing a selection of walls and a target (see Fig. 7). The perimeter of the world is walled, preventing robots from leaving the maze. Robots can see the target from any location (the target is much taller than the walls). To reach the target,

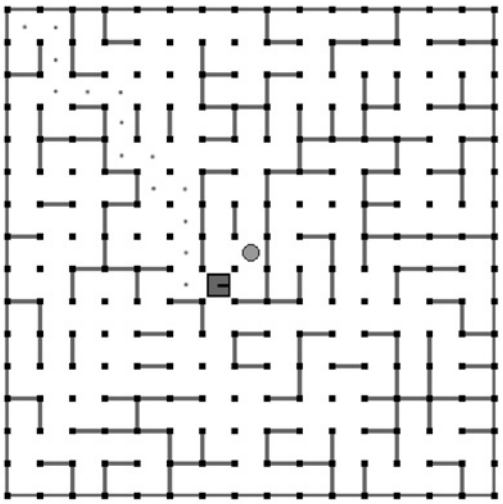

Fig. 7. Evolved maze and robot using AVA. Robot (square, facing east) starts in north-west corner and moves toward central target (circle) using evolved rules. Visited cells are marked in the center by a small dot.

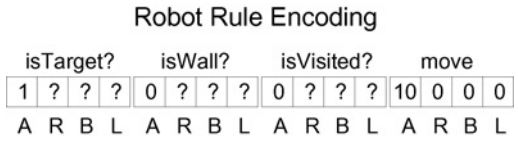

Fig. 8. Example robot rule. This can be read as: "*if* target is ahead *and* there is no wall ahead *and* ahead has not previously been visited *then* move ahead with probability 10/10."

robots must navigate around walls they encounter. Unable to plan ahead, robots react to the local state of the world (the current grid cell that they are in). Robots possess a simple memory of previously visited grid locations (consider dropping a trail of bread crumbs). Robots aim to evolve decision rules capable of navigating from a start position to the target in a previously unseen maze. This is a nontrivial task.

It has been demonstrated that this domain causes difficulties for CGAs [44]. The asymmetric bias favoring mazes allows them to outperform robots very easily. We attempt to replicate the work of [44] in order to test the efficacy of AVA in this domain. As such, [44]'s problem definition and experimental setup is closely followed.

B. Robot-Maze Coevolution

We use the following genetic representation.

1) *The Problem:* Robots are placed within a square maze consisting of 15×15 cells. Robots aim to navigate from a starting point located at the corner of a maze to a target at the center.

2) *Robots:* Robot navigation is controlled by a classifier system containing 20 mutable rules and one default rule. Each rule has 12 inputs and 4 outputs, encoded as an integer array of length 16 (see Fig. 8). Inputs are separated into three groups representing *isTarget*?, *isWall*?, and *isVisited*?. Each group contains four inputs *Ahead* (*A*), *Right* (*R*), *Behind* (*B*), and *Left* (*L*). Each input takes one of three values: 0 (*No*), 1 (*Yes*), and ? (*Don'tCare*). Input bits set to ? always evaluate true. Bits set to 0 or 1 will only evaluate true if the condition is met. Rules fire if and only if all input bits evaluate true. Hence, an input rule $\{0, ?, 1, ?\}$ for the group *isWall*? is true if there is a wall behind and no wall ahead. The default rule has all input values set to ?. Default values cannot be altered and ensure that the classifier will always fire at least one rule. To evaluate *isVisited*? each robot keeps an internal record of previously visited cells.

Robot rules are stored in an ordered list (default last) and evaluated in turn. The first rule to fire determines robot action. Each rule has four output integers, representing the probability of moving *A*, *R*, *B*, or *L*. Outputs, O_i, take values in the range $[0, 10]$, where $\sum_{i=1}^{4} O_i \leq 10$. When a rule fires an output direction is selected with probability $O_i/10$. Note that if $\sum_{i=1}^{4} O_i < 10$ there is a possibility that no direction will be selected. If no *legal* direction is selected, the rule "drops through"; remaining rules are evaluated in turn. Finally, the default rule is repeatedly evaluated until a legal direction is selected. To prevent an infinite loop, the default has an additional output constraint: $\forall i$, $O_i \geq 1$. When entering a new cell, robots face in the direction of last movement.

Robots reproduce sexually (tournament 10) using uniform crossover. Rules are copied (in entirety) from either parent with equal probability. Each rule is mutated with probability $m_{bot} = 0.01$, moving the rule one place forward in the classification order. Within each rule, bits are mutated with probability m_{bot}. For inputs, the value is randomized with equal probability. For outputs, the current value is increased or decreased by 1 (subject to constraints). Robots are initialized with all inputs set to ?.

3) *Mazes:* Each maze consists of a bit string of length 420, where 1 represents "internal wall exists." This encoding describes all possible mazes in a 15^2 grid. Mazes reproduce asexually (tournament 10); bits flipped with mutation probability $m_{maze} = 0.005$. After mutation, a breadth-first search is used to check that the newly created maze is fully connected. If not, the maze is rejected and a new offspring created. Mazes are initialized empty.

4) *Evaluation:* Robots are evaluated in pair-wise competition with mazes. The robot's journey length, j_i, from starting cell, i, to target, t, is used to calculate the robot's score, s_i. For each maze the minimum path length to target, p_i, is calculated using a breadth-first search. If $j_i \geq 2p_i$ then $s_i = 0$. If $j_i = p_i$ then $s_i = 100$. Otherwise, if $p_i < j_i < 2p_i$ then

$$s_i = 100\left(1 - \frac{j_i - p_i}{p_i}\right). \tag{20}$$

Robots are assessed from each corner, giving a sum score of $S = \sum_{i=1}^{4} s_i$. Conversely, the maze scores $m_{score} = 400 - S$.

5) *Coevolution:* 100 robots and 100 mazes coevolve for N_{gen} generations. Each generation, competitors are evaluated against six opponents. All runs are repeated N_{run} times. Let RV(v_r,v_m) denote fixed robot virulence, v_r, and fixed maze virulence, v_m. Also, set MAX = RV(1, 1) and AVA ($\alpha = 0.0125$, $\tau = 0.56$, $\mu = 0.3$, $v_0 = 0.75$). Once again, the parameter settings for AVA are the same as those selected in Section IV-D and used throughout this paper.

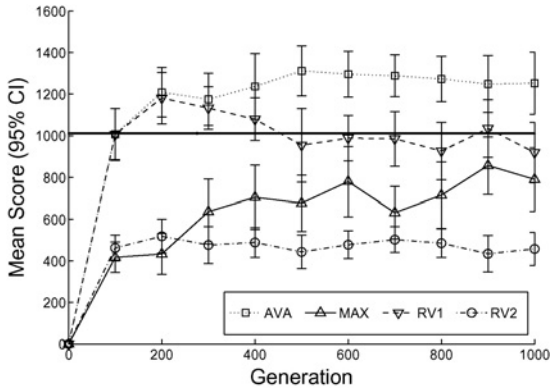

Fig. 9. Mean performance (±95% C.I.) of best evolved robots over 30 runs. The horizontal line shows the baseline performance of HDFS, a standard AI technique. AVA significantly outperforms HDFS and all fixed RV conditions.

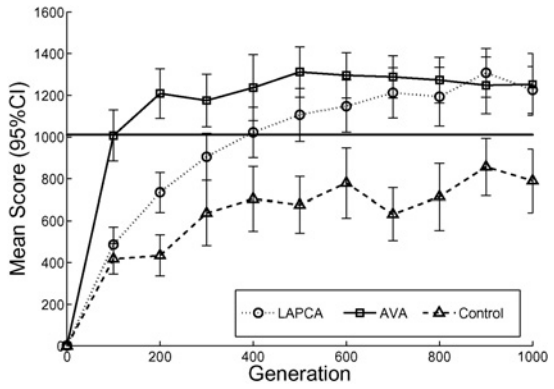

Fig. 10. Mean performance (±95% C.I.) of LAPCA over 30 runs (results for AVA and the control are reproduced from Fig. 9). The horizontal line shows the baseline performance of HDFS, a standard AI technique. AVA and LAPCA significantly outperform the control. AVA accelerates optimization, outperforming LAPCA for the first 800 generations, when both stabilize at a similar performance level. This difference is significant up to and including generation 400 (t-test, $p < 0.05$).

C. AVA vs. RV

Compare the performance of AVA and RV.

1) *Experimental Setup:* We follow the coevolutionary setup described in Section VI-B, with $N_{run} = 30$ and $N_{gen} = 1000$. Coevolution is run under four conditions: MAX, RV1(1.0, 0.5), RV2(0.5, 0.5), and AVA.

2) *Results:* Fig. 9 displays the result of coevolution. The mean scores (30 runs, ±95% CI) of the current best robot for each trial condition are graphed. Score is calculated by evaluating robots against six evaluation mazes designed to offer a range of difficulty levels. Since robot controllers are non-deterministic, scores are averaged over five repeated trials. Evaluation mazes are not exposed to robots during evolutionary learning and thus test the robustness of evolved rule sets. The horizontal line shows the performance of a standard heuristic depth-first search (HDFS) utilizing Hamming distance to target to order branches. While this HDFS is not the optimal algorithm for navigating mazes, it offers us the ability to compare the efficacy of evolved robots with a standardized technique.

MAX suffers from disengagement throughout. Although the stochastic evaluation of non-deterministic robots often preserves *some* fitness gradient (mean robot score each generation is usually greater than 0), much of this is due to noise: robots occasionally "get lucky." Thus, the system exhibits *effective* rather than full disengagement (Section II-A3).

It can be seen that RV1 outperforms MAX and RV2. By generation 200 RV1 discovers robots that are capable of navigating the evaluation mazes more successfully than HDFS. However, as time continues, RV1 robots begin to over-fit the mazes they are coevolving with. As a result, robots become less successful at navigating the evaluation mazes, with performance eventually dropping below the HDFS baseline. Nevertheless, setting $v_m = 0.5$ (RV1) offers a significant improvement over MAX. However, when v of both populations is reduced to 0.5 (RV2) progress is seriously retarded. Lacking incentive to compete, the system falls into a mediocre stable state.

In comparison, AVA outperforms the other conditions. By generation 200, AVA displays similar performance to RV1 and has discovered robots performing better than HDFS. Unlike RV1, however, AVA robots continue to outperform HDFS throughout the entire run and do not over-fit. Analysis of virulence over time shows that AVA stabilizes maze virulence at 0.54 and robot virulence at 1.0. Although this is only slightly higher than the virulence of RV1, the additional selection pressure on mazes generates enough novelty to keep robots from over-fitting (Section II-A1).

D. AVA vs. LAPCA

Following Section V-C, the performances of AVA and LAPCA are compared in this domain. However, since the computational cost of assessing navigation is much greater than assessing sorting, the setup of LAPCA has been modified. First, the robot archive is constrained using pruning (Section V-C) to a maximum size of $|\boldsymbol{A}_r|^{max} = 50$. Second, rather than submit the entire robot and maze population to the archive each generation, a sample of $S_a = 5$ individuals from each population are selected at random for archive submission. This significantly cuts down on the archive overhead. Using the notation of Section V-C2, with robot and maze sizes of $|\boldsymbol{P}_r| = |\boldsymbol{P}_m| = 100$, the minimum archive cost each generation, if all offspring are submitted is $C^{min} = 10\,000$. By submitting a random sample of $S_a = 5$ offspring, this cost falls to $C^{min} = 25$. Without this "sample submission" routine, LAPCA becomes effectively unusable for the maze navigation domain as set up here. Preliminary runs showed that the archives quickly approach and stabilize at $|\boldsymbol{A}_r| = 50$ and $|\boldsymbol{A}_r| \approx 10$. At this size, the archive requires 325 assessments each generation; an increase of 50% over the 600 assessments performed by the generator.

We use LAPCA ($N = 5$) pruned such that $|\boldsymbol{A}_r|^{max} = 50$. For direct comparison, the generator uses MAX with the same setup as Section VI-B. Parents are selected from the archive with probability 0.1.

1) *Experimental Setup:* We follow the setup described in Section VI-B, with $N_{run} = 30$ and $N_{gen} = 1000$. Coevolution is run under three conditions: MAX, AVA, and LAPCA.

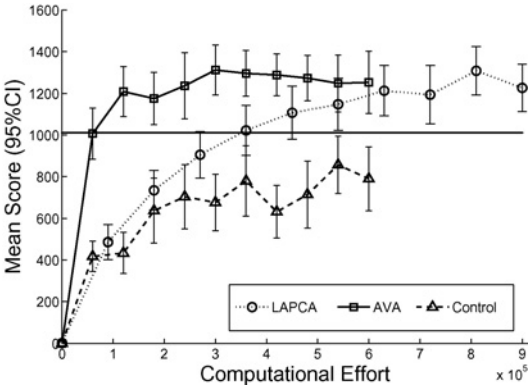

Fig. 11. Mean performance (±95% C.I., 30 runs) of best evolved robots against computational effort. AVA accelerates optimization, significantly outperforming LAPCA for the first 4×10^5 evaluations. While both stabilize at a similar level, LAPCA requires nearly five times more evaluations than AVA to reach this performance.

TABLE IV
PERFORMANCE RANKING SUMMARY

Problem	Algorithm	Performance Ranking	
		Generation	Computation
Sorting	AVA[1]	2	**1***
	LAPCA[2]	1	**3***
	RV(1, 1)	**3***	**2***
Navigation	AVA[1]	1	**1***
	LAPCA[3]	2	**2***
	RV(1, 1)	**3***	**3***

[1]AVA ($\alpha = 0.0125$, $\tau = 0.56$, $\mu = 0.3$, $v_0 = 0.75$)
[2]LAPCA (layers = 5, max archive = 250)
[3]LAPCA (layers = 5, max archive = 50, sample = 5)
*Statistically significant (95% confidence level)

2) *Results:* Figs. 10 and 11 show the results of LAPCA maze-robot coevolution. For comparison, the results of AVA and the control, MAX, are reproduced from Fig. 9. Performance is plotted against generational time. LAPCA is shown to outperform the control, producing a monotonic increase in performance until settling at a level significantly better than the HDFS baseline. We also see LAPCA underperform AVA for the first 800 generations [this difference is only significant (t-test) for generations ≤ 400 at the 0.05 level]. By generation 1000, LAPCA stabilizes at a level similar to that of AVA. Fig. 11 plots performance against computational effort (number of evaluations). AVA is shown to accelerate optimization, requiring less effort to achieve a similar performance level for the initial 4×10^5 evaluations. While both achieve a similar final performance, LAPCA requires nearly five times more evaluations than AVA to first reach this level.

These results indicate that AVA is a preferable alternative to LAPCA in this problem domain, achieving equivalent performance with lower computational cost.

VII. CONCLUSION

AVA, a novel technique designed to combat disengagement by adapting the virulence of coevolving populations during runtime, has been shown to automatically find target virulence values for the trivial GT game. By controlling bias, the GT game can be used to simulate asymmetry in real world coevolutionary problems. By solving GT across all bias settings, results suggest that AVA can effectively counter the asymmetrical bias that leads to disengagement in many real world domains.

AVA parameter settings were evolved in the GT domain over a selection of bias profiles to discover robust, domain general values. Results suggested typical AVA settings ($\alpha = 0.0125$, $\tau = 0.56$, $\mu = 0.3$, $v_0 = 0.75$), allowing AVA to be used off-the-shelf as a domain independent technique. Using these parameters, AVA was tested in the more complex domains of sorting networks and maze navigation. Table IV summarizes comparisons with LAPCA, a standard pareto archiving technique from the literature, and canonical coevolution, RV(1, 1). We see that, on a generational basis, there is no significant difference between the performance of AVA and LAPCA. However, when comparing performance against computational effort, AVA significantly outperforms LAPCA in both domains, accelerating the discovery of quality solutions and significantly reducing the computational overhead. This is a positive result, suggesting that AVA is a domain general technique that offers significant efficiency savings.

Given that AVA is specifically designed to combat disengagement, results suggest that disengagement is a greater problem in these domains than focusing or intransitivity and that, under these conditions, AVA is probably a more useful technique. Further, in domains that require lengthy assessment routines, AVA is the only realistic choice of the two.

Considering [56]'s No Free Lunch theorem, it is dangerous to suggest that AVA is better than LAPCA in general, since it is likely that there are many domains in which AVA fails to achieve the same performance. In domains where disengagement is likely, the computational efficiency of AVA suggests that it should be selected as first choice, before resorting to intensive archiving procedures. However, if a domain contains lots of intransitive superiority relationships, pareto archiving techniques such as LAPCA are likely to be a safer option. Having no evolutionary memory, AVA is susceptible to intransitivity and the cycling that may result.

An interesting avenue for future research will be to see how AVA can be best utilized on a per-dimension basis in a multiobjective domain: that is, to use multiple AVA routines, one for each dimension. Potentially, this will have the capability to overcome intransitivity and stop dimensionwise disengagement, which can lead to focusing, mediocre stability, and cycling. Perhaps this approach could be integrated into a pareto model, once the underlying problem dimensions have been exposed.

Further extensions will aim to derive a set of heuristics for fine-tuning the choice of AVA parameters in specific domains.

REFERENCES

[1] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000.
[2] R. P. Wiegand, W. C. Liles, and K. A. De Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *Proc. GECCO*, 2001, pp. 1235–1245.

[3] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D*, vol. 42, nos. 1–3, pp. 228–234, Jun. 1990.

[4] C. W. Reynolds, "Competition, coevolution and the game of tag," in *Proc. 4th Int. Workshop Simulation Synthesis Living Syst. (ALife)*, Jul. 1994, pp. 59–69.

[5] K. Sims, "Evolving 3-D morphology and behavior by competition," in *Proc. 4th Int. Workshop Simulation Synthesis Living Syst. (ALife)*, Jul. 1994, pp. 28–39.

[6] R. A. Watson and J. B. Pollack, "Symbiotic combination as an alternative to sexual recombination in genetic algorithms," in *Proc. 6th PPSN*, Sep. 2000, pp. 425–434.

[7] P. Lichodzijewski and M. I. Heywood, "Managing team-based problem solving with symbiotic bid-based genetic programming," in *Proc. GECCO*, Jul. 2008, pp. 363–370.

[8] R. A. Watson, T. Reil, and J. B. Pollack, "Mutualism, parasitism, and evolutionary adaptation," in *Proc. 7th Int. Workshop Simulation Synthesis Living Syst. (ALife)*, Aug. 2000, pp. 170–178.

[9] T. Eguchi, K. Hirasawa, J. Hu, and N. Ota, "A study of evolutionary multiagent models based on symbiosis," *IEEE Trans. Syst. Man Cybern. B*, vol. 36, no. 1, pp. 179–193, Feb. 2006.

[10] J. B. Pollack and A. D. Blair, "Co-evolution in the successful learning of a backgammon strategy," *Mach. Learning*, vol. 32, no. 3, pp. 225–240, 1998.

[11] J. Noble and R. A. Watson, "Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection," in *Proc. GECCO*, Jul. 2001, pp. 493–500.

[12] D. Cliff and G. F. Miller, "Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations," in *Proc. 3rd ECAL*, Jun. 1995, pp. 200–128.

[13] J. Cartlidge and S. Bullock, "Combatting coevolutionary disengagement by reducing parasite virulence," *Evol. Comput.*, vol. 12, no. 2, pp. 193–222, 2004.

[14] R. A. Watson and J. Pollack, "Coevolutionary dynamics in a minimal substrate," in *Proc. GECCO*, Jul. 2001, pp. 702–709.

[15] A. Bucci and J. B. Pollack, "Focusing vs. intransitivity: Geometrical aspects of co-evolution," in *Proc. GECCO*, Jul. 2003, pp. 250–261.

[16] S. G. Ficici and J. B. Pollack, "Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states," in *Proc. 6th Int. Conf. Artif. Life (Alife)*, Jun. 1998, pp. 238–247.

[17] J. Cartlidge and S. Bullock, "Caring vs. sharing: How to maintain engagement and diversity in coevolving populations," in *Proc. 7th ECAL*, Sep. 2003, pp. 299–308.

[18] X. Yao, Y. Liu, and P. Darwen, "How to make best use of evolutionary learning," in *Complex Systems: From Local Interactions to Global Phenomena*, R. Stocker, H. Jelinek, and B. Durnota, Eds. Amsterdam, The Netherlands: IOS Press, 1996, pp. 229–242.

[19] S. A. Kauffman and S. Johnsen, "Co-evolution at the edge of chaos: Coupled fitness landscapes, poised states, and co-evolutionary avalanches," *J. Theor. Biol.*, vol. 149, no. 4, pp. 467–505, 1991.

[20] L. van Valen, "A new evolutionary law," *Evol. Theory*, vol. 1, pp. 1–30, 1973.

[21] E. D. de Jong and J. B. Pollack, "Ideal evaluation from coevolution," *Evol. Comput.*, vol. 12, no. 2, pp. 159–192, 2004.

[22] A. Bucci and J. Pollack, "Order-theoretic analysis of coevolution problems: Coevolutionary statics," in *Proc. GECCO*, Jul. 2002, pp. 229–235.

[23] J. Cartlidge and S. Bullock, "Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization," in *Proc. CEC*, Jun. 2002, pp. 1420–1425.

[24] H. Juillé and J. Pollack, "Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules," in *Proc. 3rd Annu. GP Conf.*, Jul. 1998, pp. 519–527.

[25] C. D. Rosin, "Coevolutionary search among adversaries," Ph.D. dissertation, Dept. Comput. Sci., Univ. California, San Diego, Jun. 1997.

[26] J. Cartlidge and S. Bullock, "Unpicking tartan CIAO plots: Understanding irregular coevolutionary cycling," *Adaptive Behav.*, vol. 12, no. 2, pp. 69–92, 2004.

[27] R. Dawkins and J. R. Krebs, "Arms races between and within species," *Proc. Roy. Soc. London, Series* B, vol. 205, no. 1161, pp. 489–511, 1979.

[28] J. L. Shapiro, "Does data-model co-evolution improve generalization performance of evolving learners?" in *Proc. 5th PPSN*, Sep. 1998, pp. 540–549.

[29] G. Singh and K. Deb, "Comparison of multi-modal optimization algorithms based on evolutionary algorithms," in *Proc. GECCO*, Jul. 2006, pp. 1305–1312.

[30] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. Michigan, Ann Arbor, 1975.

[31] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Univ. Illinois at Urbana-Champaign, Urbana, illiGAL Rep. 95 001, May 1995.

[32] S. Wright, "The roles of mutation, inbreeding, crossbreeding and selection in evolution," in *Proc. 6th Int. Congr. Genet.*, vol. 1. 1932, pp. 356–366.

[33] D. E. Goldberg and J. J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd ICGA*, Jul. 1987, pp. 41–49.

[34] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evol. Comput.*, vol. 1, no. 2, pp. 127–149, 1993.

[35] P. Darwen and X. Yao, "Every niching method has its niche: Fitness sharing and implicit sharing compared," in *Proc. 4th PPSN*, Sep. 1996, pp. 398–407.

[36] D. Floreano and S. Nolfi, "God save the red queen! Competition in co-evolutionary robotics," in *Proc. 2nd Annu. Conf. GP*, Jul. 1997, pp. 398–406.

[37] B. Olsson, "Co-evolutionary search in asymmetric spaces," *Inform. Sci.*, vol. 133, no. 3, pp. 103–125, 2001.

[38] J. Bongard and H. Lipson, "'Managed challenge' alleviates disengagement in co-evolutionary system identification," in *Proc. GECCO*, Jun. 2005, pp. 531–538.

[39] S. G. Ficici and J. B. Pollack, "Pareto optimality in coevolutionary learning," in *Proc. 6th ECAL*, 2001, pp. 316–325.

[40] E. D. de Jong, "The incremental Pareto-coevolution archive," in *Proc. GECCO*, Jun. 2004, pp. 525–536.

[41] E. D. de Jong, "A monotonic archive for Pareto coevolution," *Evol. Comput.*, vol. 15, no. 1, pp. 61–93, 2007.

[42] G. A. Monroy, K. O. Stanley, and R. Miikkulainen, "Coevolution of neural networks using a layered Pareto archive," in *Proc. GECCO*, Jul. 2006, pp. 329–336.

[43] J. C. Tay, C. H. Tng, and C. S. Chan, "Environmental effects on the coevolution of pursuit and evasion strategies," *Genet. Programming Evolvable Mach.*, vol. 9, no. 1, pp. 5–37, 2008.

[44] P. J. Clarke, "Evolving maze-solving agents," in *Final Year Project*. Undergraduate dissertation, School Comput., Univ. Leeds, U.K., 2002 [Online]. Available: http://www.comp.leeds.ac.uk/cgi-bin/fyproj/reports/0102/clarke.pdf.gz

[45] L. Pagie and M. Mitchell, "A comparison of evolutionary and coevolutionary search," *Int. J. Comput. Intell. Applicat.*, vol. 2, no. 1, pp. 53–69, 2002.

[46] B. R. Levin, "The evolution and maintenance of virulence in microparasites," *Emerg. Infect. Dis.*, vol. 2, no. 2, pp. 93–102, 1996.

[47] D. Ashlock, S. Willson, and N. Leahy, "Coevolution and tartarus," in *Proc. CEC*, Jun. 2004, pp. 1618–1624.

[48] A. R. McIntyre and M. I. Heywood, "Toward co-evolutionary training of a multi-class classifier," in *Proc. CEC*, Sep. 2005, pp. 2130–2137.

[49] S. Bullock, J. Cartlidge, and M. Thompson, "Prospects for computational steering of evolutionary computation," in *Proc. 8th Workshop Int. Conf. Artif. Life*, Dec. 2002, pp. 131–137.

[50] J. Cartlidge, "Dynamically adapting parasite virulence to combat coevolutionary disengagement (abstract)," in *Proc. 11th Int. Conf. Simulation Synthesis Living Syst. (Alife)*, Aug. 2008, p. 757.

[51] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," *IRE WESCON Convention Rec.*, vol. 4, pp. 96–104, Aug. 1960.

[52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[53] D. Cliff and J. Bruten, "Minimal-intelligence agents for bargaining behaviors in market-based environments," HP Labs, Bristol, U.K., Tech. Rep. HPL-97-91, 1997.

[54] H. Juillé, "Incremental co-evolution of organisms: A new approach for optimization and discovery of strategies," in *Proc. 3rd ECAL*, Jun. 1995, pp. 246–260.

[55] T. Miconi and A. Channon, "The n-strikes-out algorithm: A steady-state algorithm for coevolution," in *Proc. CEC*, Jun. 2006, pp. 1639–1646.

[56] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Santa Fe Instit., Santa Fe, NM, Tech. Rep. SFI-TR-95-02-010, 1995.

John Cartlidge received the B.S. degree (first class) in artificial intelligence and mathematics in 2000, and the Ph.D. degree in computer science in 2004, both from the University of Leeds, Leeds, U.K.

He is currently a Research Associate with the University of Bristol, Bristol, U.K. For four years after his Ph.D. degree, he was with Hewlett-Packard Labs European Research Center, Bristol, the London Stock Exchange (LSE), London, U.K., and other smaller companies on commercial research projects including agent-based modeling of the LSE, statistical modeling of bidder behavior on eBay, and development of a proprietary dark liquidity exchange (patent pending). Between 2008 and 2010, he was a Research Associate with the University of Central Lancashire, Lancashire, U.K., where his research focused on evolutionary computation and finance. He is the Director and Co-Founder of Victoria.net, a private consultancy company specializing in financial software design and development.

Djamel Ait-Boudaoud received the Ph.D. degree in system and circuit integration of digital signal processing applications from the University of Nottingham, Nottingham, U.K., in 1991.

He is currently the Dean of the Faculty of Technology, University of Portsmouth, Portsmouth, U.K. He has published in the areas of signal processing architectures, evolutionary algorithms, image compression, computer vision, and dynamically reconfigurable systems. He has led and managed a number of successful industry-funded research grants and knowledge transfer programs.

Dr. Ait-Boudaoud is a Chartered Engineer and a Fellow of the Institution of Engineering and Technology.